(Http://www.qinshaoxuan.com/

Home (http://www.qinshaoxuan.com/)

- Music (http://www.ginshaoxuan.com/music)
- About (http://www.qinshaoxuan.com/about)

LeetCode 327 Count of Range Sum (http://www.qinshaoxuan.com/articles/930.html)

O2016-01-14 Q No comments (http://www.ginshaoxuan.com/articles/930.html#respond) ☑ Article

题目链接: Count of Range Sum (https://leetcode.com/problems/count-of-range-sum/)
思路来自: http://algobox.org/count-of-range-sum/ (http://algobox.org/count-of-range-sum/)
题意: 给一个数组nums,给出最大值upper,最小值lower,求有多少个区间 $[i,j](i\leqslant j)[i,j](i\leqslant j)$ 满足 $lower\leqslant\sum_{k=i}^{j}nums_k\leqslant upperlower\leqslant\sum_{k=i}^{j}nums_k\leqslant upper$

因为有负数,前缀和没有了单调性,就不能用二指针的方法。

一开始想到应该是分治,但是合并的方法没有想对。

想的是先预处理前缀和,再记录每个和的个数,然后枚举左区间的坐标,算出右区间的临界值。

但是记录和的个数就要用map,算临界值需要二分查找,时间复杂度应该是 O(nlog(n)log(n))O(nlog(n)log(n))

感觉想复杂了,于是百度了一发,发现了一个巧妙的解法。

先预处理前缀和 $sum_{i+1} = sum_i + nums_i sum_{i+1} = sum_i + nums_i$,于是区间[i,j]的和等于 $sum_j - sum_i sum_j - sum_i$ (这里和平时的写法不一样,是方便后面处理归并时的 坐标)

巧妙的地方在于,利用归并排序,使左右区间的sum有序,而且不会破坏左区间和右区间的相对位置,这样就可以用二指针了。

枚举左区间的坐标i,找到右区间中:

第一个 $sum_{rl} - sum_i sum_{rl} - sum_i$ 大于等于lower的临界值坐标rl

第一个 $sum_{rr} - sum_i sum_{rr} - sum_i$ 大于upper的临界值坐标rr

则贡献的区间个数为 rr-rlrr-rl

因为左右区间都是单调递增的,当i向右移,rl和rr也只可能向右移,所以合并时复杂度为 O(n) O(n)

计算完后进行归并排序,保证当前区间sum有序

因为只需要计算区间的个数,所以利用前缀和就可以避开不能改变位置的问题,巧妙的利用排序求解。

所以不能因为是统计区间就习惯性的排除排序这种方法啊=。=

还要注意要用long,数据的范围是int,但两个数相加就可能超出int范围

时间复杂度 O(nlog(n))O(nlog(n))

```
LeetCode 327
                                                                                                                                                      Java
1
     public class Solution {
2
         public int countRangeSum(int[] nums, int lower, int upper) {
             int ans = 0;
3
             long[] sum = new long[nums.length + 1];
4
5
             for (int i = 0; i < nums.length; ++i) {
                  sum[i + 1] = sum[i] + nums[i];
6
7
             ans = sort(sum, 0, nums.length + 1, lower, upper);
8
9
             return ans:
10
11
12
         public int sort(long[] sum, int 1, int r, int lower, int upper) {
13
             if (r - 1 \le 1) return 0;
             int ret = 0;
14
15
             int mid = (1 + r) >> 1;
16
             long[] temp = new long[r - 1];
             ret += sort(sum, 1, mid, lower, upper) + sort(sum, mid, r, lower, upper);
17
             int rl = mid, rr = mid, j = mid;
18
             for (int i = 1, k = 0; i < mid; ++i) {
19
                 if (rl < r && sum[rl] - sum[i] <= upper) {</pre>
20
                      while (rl < r && sum[rl] - sum[i] < lower) rl++;</pre>
21
22
                     while (rr < r && sum[rr] - sum[i] <= upper) rr++;</pre>
23
                      ret += rr - rl;
24
25
                  while (j < r \&\& sum[j] < sum[i]) temp[k++] = sum[j++];
                  temp[k++] = sum[i];
26
27
             System.arraycopy(temp, ∅, sum, 1, j - 1);
28
29
             return ret;
30
31
```

Categories: LeetCode (http://www.qinshaoxuan.com/articles/category/java/leetcode)

Tags: 前缀和 (http://www.qinshaoxuan.com/articles/tag/prefix-sum), 归并排序

0													最	新 最	早 :	最热
							还没有评论,	沙发等你来	抢							
社交帐	号登录:	微信	微博	QQ	人人	更多»										
轩	说点什么	么吧														
															发	fi fi
											xuan's bl	og正在使	用多说 (http	://duo	shuo	.com)
0													最	新最	早 :	最热
							还没有评论,	沙发等你来	抢							
社交帐	号登录:	微信	微博	QQ	人人	更多»										
轩	说点什么	么吧														
															发	——— 布

xuan's blog正在使用多说 (http://duoshuo.com)

- 支持使用微信、微博和QQ的账户登陆进行评论。由各自网站直接认证,不会泄露你的密码。
- 登陆后可选择分享评论到所绑定的社交网络,如微博和QQ空间。
- 评论提交后无法修改。如需修改,请删除原评论再重新提交。
- 评论支持LATEX代码,行内公式请用\(a+b=c\),行间公式请用\[a+b=c\]。公式只支持英文字符,发表评论后请刷新

© 2015-2016 xuan's blog | Theme by Theme in Progress | Proudly powered by WordPress | 蜀ICP备15025040号 (http://www.miitbeian.gov.cn/)

Admin (http://www.qinshaoxuan.com/wp-admin/) Email (http://mail.qq.com/cgi-bin/qm_share?t=qm_mailme&email=gLS1ubS0sLOzsMDx8a7j7_0) QQ (http://wpa.qq.com/msgrd?v=3&uin=459440330&site=qq&menu=yes)

返回顶部 关于我 (http://www.qinshaoxuan.com/about) 发表评论 前往底部